Privilege Escalation

Course Introduction

Alexis Ahmed

Senior Penetration Tester @HackerSploit

Offensive Security Instructor @INE

Course Topic Overview

- Introduction To Privilege Escalation
- Windows Privilege Escalation Techniques
  - Privilege Escalation Scripts (PowerUp.ps1 etc)
  - Locally Stored Credentials
  - Insecure Service Permissions
  - Registry AutoRun
  - Bypassing UAC
  - Impersonation Attacks
  - DLL Hijacking
- Linux Privilege Escalation Techniques
  - Locally Stored Credentials
  - Misconfigured File Permissions
  - SUID Binaries
  - Misconfigured SUDO Permissions
  - Shared Library Injection

Prerequisites

- Basic Understanding of Computer Networking
  - Knowledge of IP addresses, subnetting, routing, and network devices (switches, routers, firewalls).
  - Familiarity with common network protocols (TCP, UDP, HTTP, DNS, etc.).
- Fundamentals of Operating Systems
  - Basic knowledge of Windows and Linux operating systems, including their command-line interfaces.
  - Understanding of system processes, file systems, and user permissions.
- Fundamental Knowledge & Experience in Exploitation & Post-Exploitation
  - Experience in performing local enumeration on Windows and Linux targets.
  - Experience with the process of exploitation and post-exploitation on Windows and Linux.
- Experience with Penetration Testing Tools
  - Some experience using common penetration testing tools (e.g., Metasploit, Nmap etc).
  - Understanding of basic penetration testing methodologies and techniques.

Learning Objectives:

1. Understand the Concept of Privilege Escalation
   - Define privilege escalation and explain its importance in penetration testing and red teaming.
2. Identify Common Vulnerabilities Leading to Privilege Escalation
   - Describe common misconfigurations and security flaws in Windows and Linux environments.
   - Recognize typical attack vectors used to escalate privileges.
3. Conduct System Enumeration to Gather Critical Information
   - Utilize various tools to enumerate Windows and Linux systems for valuable information.
   - Interpret and analyze system data to identify potential privilege escalation opportunities.
4. Apply Techniques to Escalate Privileges on Windows Systems
   - Demonstrate the use of insecure services, token impersonation, and other techniques for privilege escalation on Windows.
   - Utilize tools like Metasploit, PowerShell, and PowerUp for privilege escalation.
5. Apply Techniques to Escalate Privileges on Linux Systems
   - Exploit SUID binaries, misconfigured SUDO permissions, and other common Linux privilege escalation methods.
6. Leverage Advanced Privilege Escalation Techniques
   - Implement advanced techniques such as DLL hijacking, bypassing UAC shared object injection.

Let’s Get Started!

Introduction To Privilege Escalation

Privilege Escalation

- Privilege escalation is a critical concept in penetration testing and red teaming.
- It refers to the process of gaining elevated access or additional privileges in a computer system or network, typically from a lower-level user to a higher-level user or administrator.
- Privilege escalation involves exploiting vulnerabilities or misconfigurations to gain access to resources that are typically restricted to users with higher privileges.

Types of Privilege Escalation

Privilege escalation can be divided in two types: vertical and horizontal.

- Vertical: the attacker is able to move from a lower privileged user to a higher privileged user. For example from a low-end user to administrator or root user.

- Horizontal: the attacker keeps the same set or level of privileges, but assumes the identity of a different user (he/she does not gain any further privilege).

Types of Privilege Escalation

To make things a bit clearer, here are some examples.

- Vertical privilege escalation: on a Linux OS, the attacker is able to escalate privileges from a user (i.e. applications) and gain root privileges.

- Horizontal: on a Windows OS, the attacker is able to assume the identity on any other Standard user on the system. The attacker is not escalating privileges from a Standard user to an Administrator user.

Privilege Escalation With PowerUp

Privilege Escalation With PowerUp

- PowerUp is a popular tool used in the context of Windows privilege escalation.
- It is part of the PowerSploit framework, a collection of PowerShell-based tools designed for offensive security tasks, including enumeration, exploitation, and post-exploitation activities.
- PowerUp specifically focuses on identifying common privilege escalation vectors within a Windows environment.

Privilege Escalation With PowerUp

- PowerUp automates the process of scanning a Windows system for potential misconfigurations, vulnerabilities, and security flaws that could lead to privilege escalation. It performs a comprehensive set of checks to identify opportunities for privilege escalation, such as:

- Insecure Service Configurations: Services running with elevated privileges (e.g., SYSTEM) that are vulnerable to exploitation due to weak permissions or other security issues.
- Unquoted Service Paths: Services with unquoted paths that can be exploited by placing a malicious executable in a strategic location.
- Weak Registry Permissions: Registry keys with insecure permissions that allow unauthorized modification, leading to privilege escalation.

Privilege Escalation With PowerUp

- Vulnerable Scheduled Tasks: Scheduled tasks that can be manipulated to run with elevated privileges.

- Insecure File Permissions: Files or directories with weak permissions that could be exploited to execute code with higher privileges.

- Insecure DLL Search Orders: Exploitable DLL search orders that allow DLL hijacking to gain elevated privileges.

- Stored Credentials: Credentials stored insecurely in registry keys, files, or other locations.

Lab Demo: Privilege Escalation With PowerUp

References & Resources

- PowerUp: https://github.com/PowerShellMafia/PowerSploit/blob/master/Privesc/PowerUp.ps1

Privilege Escalation With PrivescCheck

Lab Demo: Privilege Escalation With PrivescCheck

Unattended Installation Files

Windows Configuration Files

- Windows can automate a variety of repetitive tasks, such as the mass rollout or installation of Windows on many systems.
- This is typically done through the use of the Unattended Windows Setup utility, which is used to automate the mass installation/deployment of Windows on systems.
- This tool utilizes configuration files that contain specific configurations and user account credentials, specifically the Administrator account's password.
- If the Unattended Windows Setup configuration files are left on the target system after installation, they can reveal user account credentials that can be used by attackers to authenticate with Windows target legitimately.

Unattended Windows Setup

- The Unattended Windows Setup utility will typically utilize one of the following configuration files that contain user account and system configuration information:
    - C:\Windows\Panther\Unattend.xml
    - C:\Windows\Panther\Autounattend.xml

- As a security precaution, the passwords stored in the Unattended Windows Setup configuration file may be encoded in base64.

Lab Demo: Unattended Installation Files

Windows Credential Manager

Windows Credential Manager

- Windows Credential Manager is a built-in feature in Microsoft Windows that allows users to securely store and manage their credentials, such as usernames, passwords, and other login information, for various services, applications, websites, and network resources.

- It's a part of the broader Windows security ecosystem designed to streamline authentication processes and reduce the need to repeatedly enter credentials.

cmdkey

- **cmdkey** is a command-line utility in Windows that interacts with Windows Credential Manager.

- It allows you to manage the credentials stored in the Credential Manager from the command line, offering flexibility and automation in handling credentials.

- With cmdkey, you can:
  - Add Credentials
  - List Credentials
  - Delete Credentials

Lab Demo: Windows Credential Manager

PowerShell History

Lab Demo: PowerShell History

Exploiting Insecure Service Permissions

Windows Services

- Windows services are background processes that run in the Windows operating system, often with elevated privileges.
- Services can be configured to start automatically, manually, or be triggered by specific events.
- Many critical system services run with high privileges, such as LocalSystem, LocalService, or NetworkService.

Identifying Insecure Service Permissions

- Insecure service permissions occur when a Windows service has misconfigurations in its access control settings, allowing unprivileged users to modify the service or its associated components. Common permission misconfigurations include:

    - Full Control or Write Permissions: If a service's configuration can be modified by an unprivileged user, this creates a security risk. An attacker could change the service's properties, such as its executable path, to execute arbitrary code with elevated privileges.
    - Unquoted Service Paths: If a service's executable path contains spaces and is not properly enclosed in quotes, an attacker can place a malicious executable in a specific location along the path, causing it to be executed when the service starts.

Exploiting Insecure Service Permissions

1 - Identify Vulnerable Services

- The attacker enumerates the Windows services on the target system to find those with insecure permissions. Tools like PowerUp, AccessChk, or Metasploit can be used to automate this process.

2 - Analyze Service Permissions

- The attacker checks the permissions on each service to determine if they allow unauthorized modifications.
- This involves examining the service's security descriptor and ACLs (Access Control Lists) to see who has write or full control.

Exploiting Insecure Service Permissions

3 - Modify the Service Configuration

- If a service has insecure permissions, the attacker can modify its properties. For example, the attacker might change the ImagePath to point to a malicious executable, allowing them to execute code with the service's privileges.

4 - Restart the Service

- Once the service configuration has been modified, the attacker restarts the service. This causes the modified executable to run, leading to privilege escalation. If the service runs with administrative or system-level privileges, the attacker's code will execute with those privileges.

Service Binary Replacement

1 - Identify a Service with Insecure Permissions

- Attackers first find a service where they have write access to the service's executable file or its directory.

2 - Replace the Executable

- The attacker replaces the existing service executable with a malicious executable. The malicious executable might contain a payload to create backdoors, escalate privileges, or perform other unauthorized actions.

3 - Restart the Service

- When the service is restarted, it runs the malicious executable, allowing the attacker to execute their payload with the privileges of the service, often resulting in privilege escalation.

Lab Demo: Exploiting Insecure Service Permissions

Privilege Escalation Via Registry AutoRuns

Registry AutoRun

- Registry autoruns are a common vector for privilege escalation in Windows systems.
- This technique involves exploiting registry keys that are used to configure programs or scripts to automatically run when certain events occur, such as system startup, user login, or service initialization.
- Attackers can leverage insecure configurations or weak permissions in these registry keys to execute malicious code with elevated privileges.

Registry AutoRun

- Windows uses specific registry keys to define which programs or scripts should automatically run under certain conditions. Common autorun scenarios include:
  - System Startup: Programs configured to start when the system boots.
  - User Login: Programs that run when a specific user logs in.
  - Service Initialization: Services that start with the system and potentially with elevated privileges.

Privilege Escalation Via Registry AutoRun

1 - Identify Vulnerable Registry Autoruns

- The attacker identifies registry keys that control autoruns and checks their permissions. Tools like AccessChk or PowerUp can help locate insecure keys.

2 - Exploiting Weak Permissions

- If the attacker has write access to a registry key used for autoruns, they can modify the key's value to point to a malicious executable or script. This executable will then run with the permissions of the original autorun process, often leading to privilege escalation.

3 - Achieving Privilege Escalation

- When the system restarts or the target user logs in, the malicious code runs with elevated permissions, granting the attacker higher privileges or allowing them to perform unauthorized actions.

Common Registry Keys For AutoRuns

Typical registry keys associated with autoruns include:

- Autoruns for system startup:
  `HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run`

- Autoruns for user login:
  `HKEY_CURRENT_USER\SOFTWARE\Microsoft\Windows\CurrentVersion\Run`

- Configuration for Windows services, which can run with elevated privileges:
  `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services`

Lab Demo: Privilege Escalation Via Registry AutoRuns

Access Token Impersonation

Windows Access Tokens

- Windows access tokens are a core element of the authentication process on Windows and are created and managed by the Local Security Authority Subsystem Service (LSASS).

- A Windows access token is responsible for identifying and describing the security context of a process or thread running on a system. Simply put, an access token can be thought of as a temporary key akin to a web cookie that provides users with access to a system or network resource without having to provide credentials each time a process is started or a system resource is accessed.

- Access tokens are generated by the winlogon.exe process every time a user authenticates successfully and includes the identity and privileges of the user account associated with the thread or process. This token is then attached to the userinit.exe process, after which all child processes started by a user will inherit a copy of the access token from their creator and will run under the privileges of the same access token.

Windows Access Tokens

- Windows access tokens are categorized based on the varying security levels assigned to them. These security levels are used to determine the privileges that are assigned to a specific token.

- An access token will typically be assigned one of the following security levels:
  - Impersonate-level tokens are created as a direct result of a non-interactive login on Windows, typically through specific system services or domain logons.
  - Delegate-level tokens are typically created through an interactive login on Windows, primarily through a traditional login or through remote access protocols such as RDP.

- Impersonate-level tokens can be used to impersonate a token on the local system and not on any external systems that utilize the token.

- Delegate-level tokens pose the largest threat as they can be used to impersonate tokens on any system.

Windows Privileges

- The process of impersonating access tokens to elevate privileges on a system will primarily depend on the privileges assigned to the account that has been exploited to gain initial access as well as the impersonation or delegation tokens available.

- The following are the privileges that are required for a successful impersonation attack:
  - SeAssignPrimaryToken: Allows a process to assign the primary token for a process. This privilege is critical for creating processes with specific user security contexts.
  - SeCreateToken: The SeCreateToken privilege allows a process to create new security tokens, typically used for authentication and identity representation.
  - SeImpersonatePrivilege: The SeImpersonatePrivilege allows a process to impersonate other users' security tokens. This privilege is crucial for operations where a service or process needs to perform tasks on behalf of a different user.

The Incognito Module

- Incognito is a built-in meterpreter module that was originally a standalone application that allows you to impersonate user tokens after successful exploitation.

- We can use the incognito module to display a list of available tokens that we can impersonate.

Lab Demo: Access Token Impersonation

Juicy Potato

Juicy Potato

- Juicy Potato is a Windows privilege escalation exploit that leverages specific vulnerabilities related to DCOM (Distributed Component Object Model) and the way Windows manages the communication between processes and services.

- It primarily targets Windows' token manipulation features to achieve privilege escalation from a lower-privilege user to a system-level or administrative-level user.

Juicy Potato

- Juicy Potato is based on a technique called "Potato," which involves exploiting the behavior of DCOM and the RpcSs (Remote Procedure Call Subsystem) service.

- The exploit leverages Windows' capability to create LocalService/NetworkService tokens and then use those tokens to impersonate higher-privilege accounts like SYSTEM.

How it works

DCOM and CLSIDs:

- Windows DCOM uses CLSIDs (Class Identifiers) to manage communication between different software components. When a request is made to DCOM, it can lead to the creation of new processes running under specific security contexts.

Manipulating LocalService Tokens:

- Juicy Potato exploits a vulnerability in how DCOM processes and services interact, particularly when creating tokens. It does this by leveraging the LocalService token to get access to a higher-privilege context.

Creating a Malicious COM Server:

- The exploit creates a fake COM server and registers it with a specific CLSID. This allows the attacker to direct requests to their malicious COM server, enabling them to manipulate the token used for that process.

How it works

Impersonation and Token Duplication:

- Once the malicious COM server is registered and the process initiated, Juicy Potato can create a LocalService token and then manipulate it to impersonate a high-privilege context like SYSTEM.

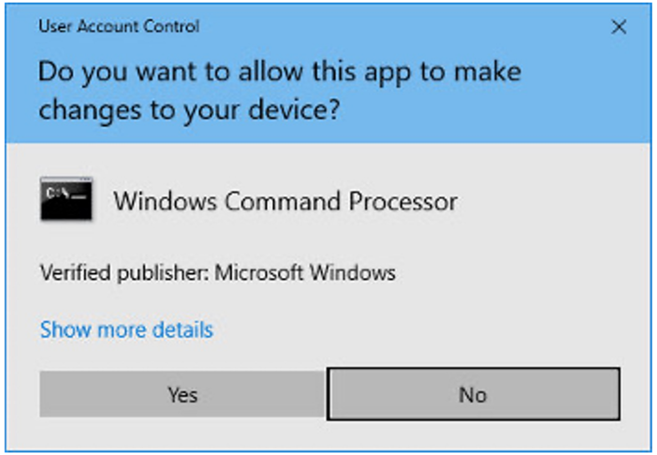
- By duplicating and adjusting the token, the exploit achieves privilege escalation.

Lab Demo: Juicy Potato

Bypassing UAC With UACMe

UAC (User Account Control)

- User Account Control (UAC) is a Windows security feature introduced in Windows Vista that is used to prevent unauthorized changes from being made to the operating system.
- UAC is used to ensure that changes to the operating system require approval from the administrator or a user account that is part of the local administrators group.
- A non-privileged user attempting to execute a program with elevated privileges will be prompted with the UAC credential prompt, whereas a privileged user will be prompted with a consent prompt.
- Attacks can bypass UAC in order to execute malicious executables with elevated privileges.


User Account Control

Do you want to allow this app to make changes to your device?

Windows Command Processor

Verified publisher: Microsoft Windows

Bypassing UAC

- In order to successfully bypass UAC, we will need to have access to a user account that is a part of the local administrators group on the Windows target system.
- UAC allows a program to be executed with administrative privileges, consequently prompting the user for confirmation.
- UAC has various integrity levels ranging from low to high, if the UAC protection level is set below high, Windows programs can be executed with elevated privileges without prompting the user for confirmation.
- There are multiple tools and techniques that can be used to bypass UAC, however, the tool and technique used will depend on the version of Windows running on the target system.

Bypassing UAC With UACMe

- UACMe is an open source, robust privilege escalation tool developed by @hfire0x. It can be used to bypass Windows UAC by leveraging various techniques.
  - GitHub: https://github.com/hfiref0x/UACME

- The UACME GitHub repository contains a very well documented list of methods that can be used to bypass UAC on multiple versions of Windows ranging from Windows 7 to Windows 10.

- It allows attackers to execute malicious payloads on a Windows target with administrative/elevated privileges by abusing the inbuilt Windows AutoElevate tool.

- The UACMe GitHub repository has more than 60 exploits that can be used to bypass UAC depending on the version of Windows running on the target.

Lab Demo: Bypassing UAC With UACMe

DLL Hijacking

DLL Hijacking

- DLL hijacking is a privilege escalation technique, where an attacker manipulates the way Windows applications load Dynamic Link Libraries (DLLs) to execute malicious code with elevated privileges.

- This technique exploits the default search order for DLLs in Windows to replace or inject custom DLLs into an application, leading to privilege escalation.

What is DLL Hijacking?

- DLL hijacking occurs when an attacker is able to control or influence which DLLs a Windows application loads at runtime.

- Windows applications rely on DLLs for various functions, such as accessing system resources, performing tasks, or providing application-specific features.

- By manipulating the DLL loading process, an attacker can inject malicious code into an application, potentially leading to privilege escalation.

How DLL Hijacking Works

Default DLL Search Order

- Windows applications use a default search order to locate and load DLLs at runtime.
- This order includes several locations, such as the application's directory, system directories, and system-defined paths. If a DLL is not found in the expected location, Windows moves through the search order until it finds the required DLL.

Exploiting Uncontrolled DLL Loading

- An attacker can exploit this search order by placing a malicious DLL in a location where it will be loaded instead of the legitimate DLL.
- If the application runs with elevated privileges, this can lead to privilege escalation.

DLL Hijacking Methodology

Step 1: Identify Vulnerable Applications

- Determine Privileged Applications:
  - Identify applications or services that run with elevated privileges (e.g., administrator or SYSTEM).
- Analyze DLL Dependencies:
  - Examine the application's dependency on specific DLLs and check for cases where these DLLs may not be found in their expected locations.

DLL Hijacking Methodology

Step 2: Examine the DLL Search Order

- Understand Default Search Order: Windows has a predefined order for searching for DLLs. It generally starts with the application's directory, followed by the system directories, and then other system-defined paths.

- Identify Potential Insertion Points: Determine where in the search order an attacker might place a DLL so that it gets loaded by the application. Common locations include:
    - The application's current working directory.
    - The System32 or SysWOW64 directories.
    - Directories listed in the PATH environment variable.
    - Other directories included in the search order.

DLL Hijacking Methodology

Step 3: Inject a Malicious DLL

- Create a Malicious DLL: The attacker creates a DLL with the same name as a DLL that is missing or not found by the application. This DLL contains the attacker's code or payload.

- Place the Malicious DLL in a Strategic Location: Place the DLL in a location where the application is likely to look for it (search order). The goal is for the application to find and load the malicious DLL instead of the legitimate one.

Lab Demo: DLL Hijacking

Locally Stored Credentials

Lab Demo: Locally Stored Credentials

Misconfigured File Permissions

Lab Demo: Misconfigured File Permissions

Exploiting SUID Binaries

Exploiting SUID Binaries

- In addition to the three main file access permissions (read, write and execute), Linux also provides users with specialized permissions that can be utilized in specific situations. One of these access permissions is the SUID (Set Owner User ID) permission.

- When applied, this permission provides users with the ability to execute a script or binary with the permissions of the file owner as opposed to the user that is running the script or binary.

- SUID permissions are typically used to provide unprivileged users with the ability to run specific scripts or binaries with "root" permissions. It is to be noted, however, that the provision of elevate privileges is limited to the execution of the script and does not translate to elevation of privileges, however, if improperly configured unprivileged users can exploit misconfigurations or vulnerabilities within the binary or script to obtain an elevated session.

Exploiting SUID Binaries

- This is the functionality that we will be attempting to exploit in order to elevate our privileges, however, the success of our attack will depend on the following factors:
  - Owner of the SUID binary – Given that we are attempting to elevate our privileges, we will only be exploiting SUID binaries that are owned by the “root” user or other privileged users.
  - Access permissions – We will require executable permissions in order to execute the SUID binary.

Lab Demo: Exploiting SUID Binaries

Misconfigured SUDO Privileges

Lab Demo: Misconfigured SUDO Privileges

Shared Library Injection

Shared Library

- In Linux, a shared library (also known as a dynamic library or dynamic shared object, typically with a .so extension) is a file that contains code and data that can be loaded by multiple processes at runtime.

- Shared libraries allow code to be modular, reusable, and reduce memory usage, as multiple processes can use the same shared code.

Shared Library Injection

- Shared library injection involves injecting a custom shared library into a running process to execute arbitrary code or manipulate the process's behavior.
- This technique can be used for various purposes, such as debugging, monitoring, or, in the context of privilege escalation, executing code with higher privileges.

How Shared Library Injection Works

1 - Identify a Target Process

- The attacker identifies a running process with elevated privileges, such as a system service, daemon, or application running as root.

2 - Create a Malicious Shared Library

- The attacker creates a shared library containing the code they wish to execute. This code can include arbitrary payloads, backdoors, or other malicious activities designed to achieve privilege escalation.

How Shared Library Injection Works

3 - Inject the Shared Library into the Target Process

- Several techniques can be used to inject a shared library into a running process:
    - Using LD_PRELOAD: This environment variable specifies a shared library to be loaded before any other libraries. By setting this variable, an attacker can preload a malicious shared library into a process.
    - Process Control (ptrace): The ptrace system call allows a process to control another process, typically used for debugging. Attackers can use ptrace to inject code into a running process, causing it to load a malicious shared library.

Lab Demo: Shared Library Injection

Privilege Escalation

Course Conclusion

Learning Objectives:

1. Understand the Concept of Privilege Escalation
   - Define privilege escalation and explain its importance in penetration testing and red teaming.
2. Identify Common Vulnerabilities Leading to Privilege Escalation
   - Describe common misconfigurations and security flaws in Windows and Linux environments.
   - Recognize typical attack vectors used to escalate privileges.
3. Conduct System Enumeration to Gather Critical Information
   - Utilize various tools to enumerate Windows and Linux systems for valuable information.
   - Interpret and analyze system data to identify potential privilege escalation opportunities.
4. Apply Techniques to Escalate Privileges on Windows Systems
   - Demonstrate the use of insecure services, token impersonation, and other techniques for privilege escalation on Windows.
   - Utilize tools like Metasploit, PowerShell, and PowerUp for privilege escalation.
5. Apply Techniques to Escalate Privileges on Linux Systems
   - Exploit SUID binaries, misconfigured SUDO permissions, and other common Linux privilege escalation methods.
6. Leverage Advanced Privilege Escalation Techniques
   - Implement advanced techniques such as DLL hijacking, bypassing UAC and shared object injection.

Thank You!

EXPERTS AT MAKING YOU AN EXPERT

INE